TP PROGRAMMATION PCDUINO EN C 02 SOCKET

Objectifs : Etre capable de programmer en C un PCDUINO pour envoyer des requètes POST sur le réseau intranet

ok testé

1. MATÉRIEL :

PCDUINO en mode ssh en wifi par vnc ou en direct avec un écran/clavier/souris. (voir Démarrage rapide d'un pcduino)

Geany installé sur le pcduino. (Pour l'installation des paquets geany voir internet)

2. INTRODUCTION

On souhaite envoyer une requête POST vers le serveur de base de données de l'intranet.

3. REQUÊTE POST RAPPEL

Les requêtes POST permettent d'envoyer des données entre un client et un serveur. Les données sont

4. SOCKET : RAPPEL

Les sockets servent à communiquer entre deux hôtes appelés Client / Serveur à l'aide d'une adresse IP et d'un port.
SOCKET = IP + PORT
Ces sockets permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux (le client et le serveur), soit de manière fiable à l'aide du protocole TCP/IP, soit non fiable mais plus rapide avec le protocole UDP.
Nous allons utiliser le premier mode, le mode TCP/IP.

5. FONCTION GUIDE "CONNECT()"

Extrait de : PROGRAMMATION RÉSEAU /Arnaud Sangnier /sangnier@liafa.univ-paris-diderot.fr

Création d'une socket

- La création d'une socket se fait grâce à :
  - **#include <sys/socket.h >**
    int socket(int domaine, int type, int protocol)
- Pour nous :
  - **domaine** vaudra **PF_INET** (pour IPv4) ou **PF_INET6** (pour IPv6)
  - **type** vaudra **SOCK_STREAM** (pour les sockets TCP)
  - **protocol** spécifie le procole de communication (mais pour TCP, on peut mettre 0 et le protcole est chosi de façon automatique)
- L'entier renvoyé sera le descripteur utilisé pour comuniquer

Schéma Client-Serveur en C

Pour communiquer

- On va envoyer et recevoir des caractères sur le descripteur de socket
- Pour recevoir on va utiliser
- **ssize_t read(int filedes, void *buf, size_t nbyte);**
  - Remplit le buffer **buf**
  - **nbyte** est la taille maximale de **buf**
  - renvoie le nombre de données reçu (-1 si erreur et 0 si la connexion est fermée)
- Pour envoyer on va utiliser
- **ssize_t write(int filedes, void *buf, size_t nbyte);**
  - Même principe que readl en est la taille en octet de buf
  - Pour le dernier argument, si on est en IPv4 et que adresse est de type **struct sockaddr_in**, on pourra mettre **sizeof(struct sockaddr_in)**
  - Quand on a fini la communication, on peut fermer le descripteur de socket avec la commande
    - **int close(int fildes);**

Exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>

int main() {
  struct sockaddr_in adress_sock;
  adress_sock.sin_family = AF_INET;
  adress_sock.sin_port = htons(4242);
  inet_aton("127.0.0.1",&adress_sock.sin_addr);

  int descr=socket(PF_INET,SOCK_STREAM,0);
  int r=connect(descr,(struct sockaddr *)&adress_sock,
                sizeof(struct sockaddr_in));
  if(r!=-1){
    char buff[100];
    int size_rec=read(descr,buff,99*sizeof(char));
    buff[size_rec]='\0';
    printf("Caracteres recus : %d\n",size_rec);
    printf("Message : %s\n",buff);
    char *mess="SALUT!\n";
    write(descr,mess,strlen(mess));
  }
  return 0;
}
```

Autre exemple de Client TCP en mode connecté (*extrait : Programmation réseaux v1.0*)

Le mode connecté (client TCP)

● Créer le point de communication :

socket() int socketDialogue = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);

● Déterminer le point de rencontre distant (le serveur) :

remplir une structure sockaddr_in

struct sockaddr_in pointDeRencontreDistant;
socklen_t longueurAdresse;
longueurAdresse = sizeof(pointDeRencontreDistant);
memset(&pointDeRencontreDistant, 0x00, longueurAdresse);
pointDeRencontreDistant.sin_family = PF_INET; pointDeRencontreDistant.sin_port = htons(IPPORT_USERRESERVED); inet_aton("192.168.52.83", &pointDeRencontreDistant.sin_addr);

● Se connecter au point de rencontre distant (le serveur) : connect()

connect(socketDialogue, (struct sockaddr *)&pointDeRencontreDistant, longueurAdresse);

●Echanger des données avec le point de rencontre distant (le serveur) :

send()/recv() ou read()/write()

en (définissant et) utilisant un protocole et une procédure d'échange communs

● Fermer la connexion et le point de communication : close() ou shutdown()

close(socketDialogue);

A partir de ces exemples nous créons notre programme.

6. PROGRAMMATION DU SOCKET

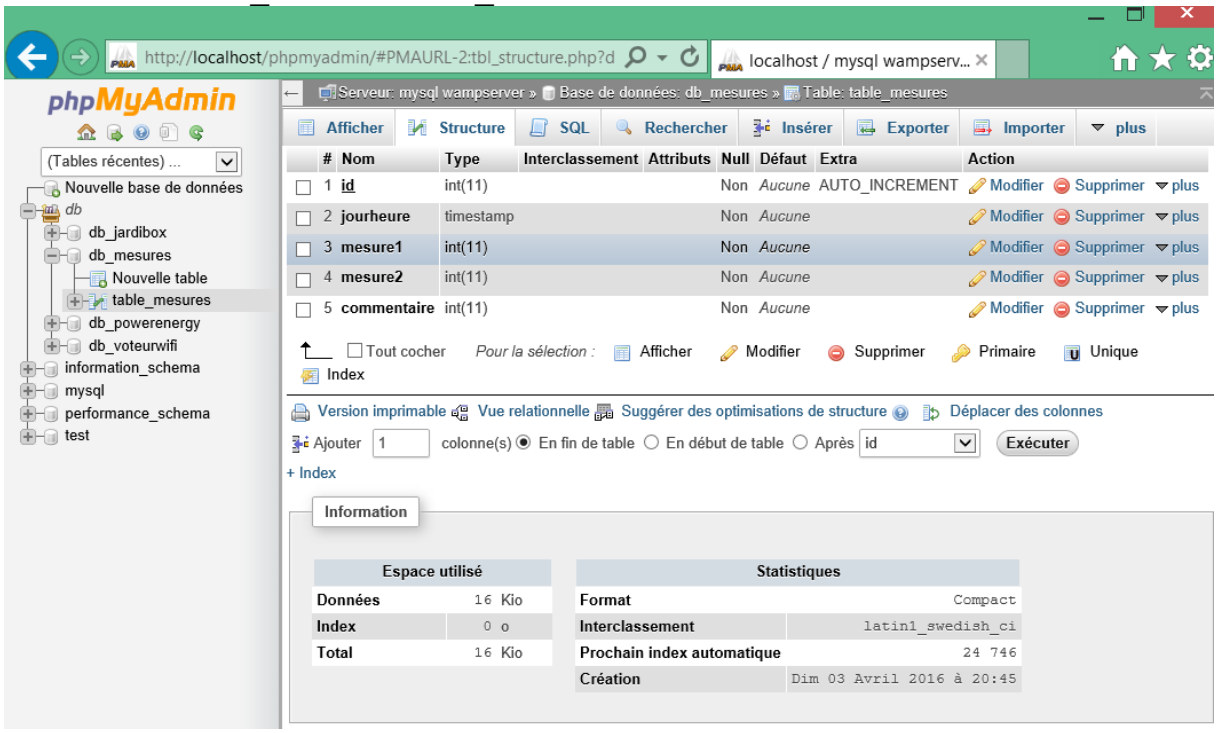
```
struct sockaddr_in adress_sock;//adresse socket avec type utilisé par connect()
adress_sock.sin_family = AF_INET;//domaine IPV4 = AF_INET
adress_sock.sin_port = htons(80);//port 80
inet_aton("192.168.1.11",&adress_sock.sin_addr);// manipulation d'adresse : IP du serveur en local
int monsocket = socket(PF_INET,SOCK_STREAM,0);
int r=connect(monsocket,(struct sockaddr *)&adress_sock,sizeof(struct sockaddr_in));

if (r!=1)//si connexion ok
{
        //envoi de la requete POST
        write (monsocket,messComplet,strlen(messComplet));
}//if
```

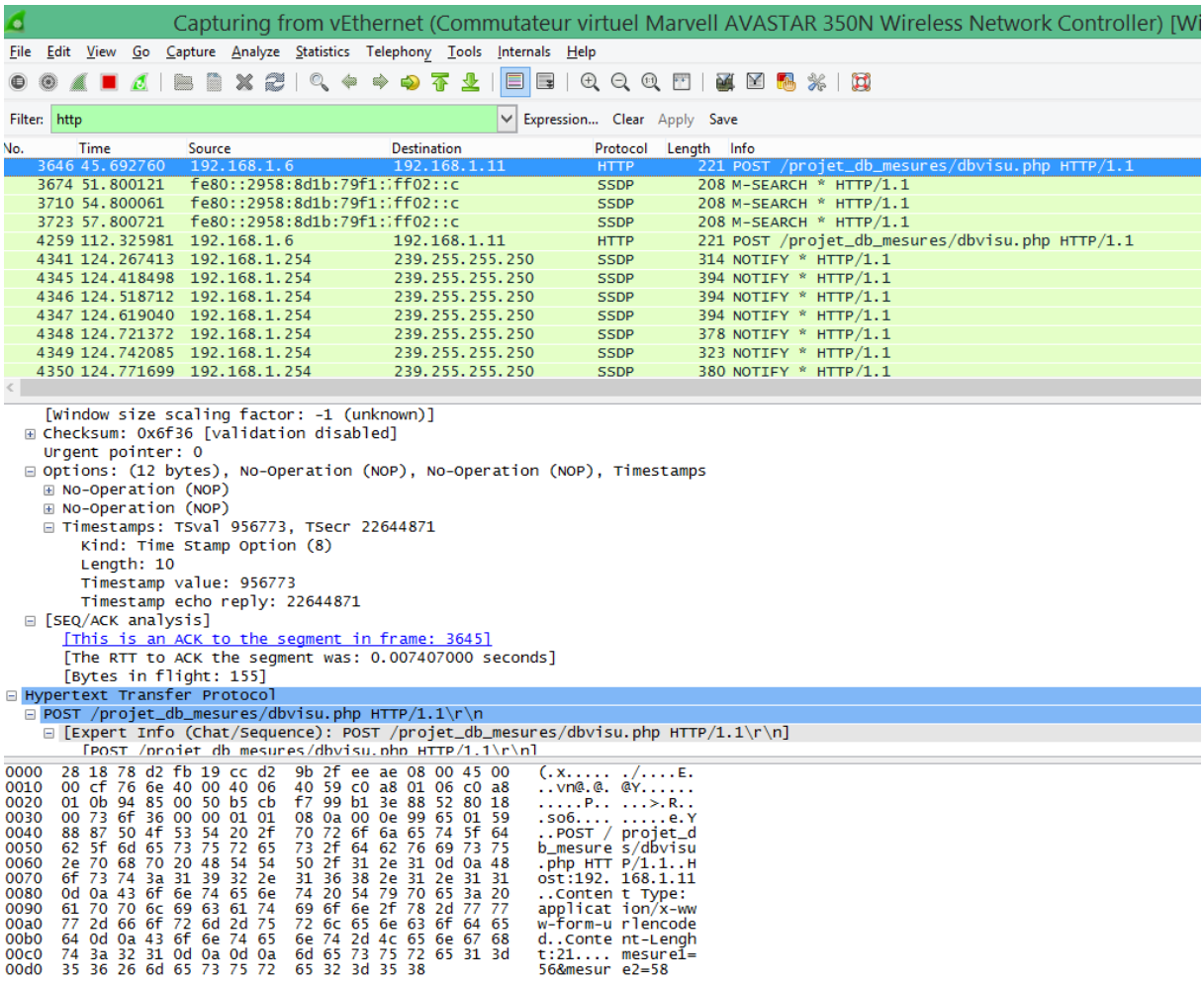
7. TEST DU FONCTIONNEMENT

Afin de tester le bon envoi du socket on utilise "wireshark".
Les conditions de test sont :

- Programme en C sur pcduino (192.168.1.39) avec geany (IDE C)
- Serveur Wamp sur PC (192.168.1.11:80).
- Base de donnée : projet_db_mesures/dbvisu.php visualise les données de la base db_mesures/table_mesures



8. MESURE AU SNIFFER "WIRESHARK"



Le POST apparaît dans l'enregistrement du sniffer.
Cette méthode permet de détecter d'éventuelles erreurs de connexion et de corriger le programme.

9. PROBLÈMES RENCONTRÉS

Problème de synthaxe de la requête :

1. erreur : Content-lenght : au lieu de Content-length !!!! (2 jours pour vérifier)

10. AMÉLIORATION DE LA ROBUSTESSE DU PROGRAMME

Prévoir le "else" si la connexion ne se fait pas

Prévoir une acquisition des données sur deux canaux analogiques (ou un analogique et un numérique
Prévoir d'écrire mes mesures dans un fichier
Prévoir d'écrire un fichier log
Prévoir de lancer le programme au démarrage

11. CORRECTION

11.1. Envoi requête POST du pcduino vers serveur web

Programme écrit avec Geany sous lubuntu sur un pcduino

Fichier projet progc.geany

```
[indentation]
indent_width=4
indent_type=1
indent_hard_tab_width=8
detect_indent=false
detect_indent_width=false
indent_mode=2

[project]
name=progc
base_path=/home/ubuntu/projects/progc/
description=
file_patterns=

[long line marker]
long_line_behaviour=1
long_line_column=72

[files]
current_page=1
FILE_NAME_0=1140;C;0;16;1;1;0;/home/ubuntu/projects/progc/essai00/main.c;0;4
FILE_NAME_1=240;C;0;16;1;1;0;/home/ubuntu/projects/progc/essai00/envoiPOST.c;0;4

[VTE]
last_dir=/home/ubuntu
```

EnvoiPOST.c

```
/*envoi requete POST à un serveur en 192.168.1.11 by SB.
//attention pour utliser modifier le messPOST en fonction de votre base de donnée
et de votre fichier php.
Modifier aussi l'IP du serveur( dans messPOST et aussi dans inet_aton
*/



#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

#include <netinet/in.h> //contient les macro de conversion de données entre les différentes machines (les données ne sont pas representées identiquement entre linux et windows...)
#include <arpa/inet.h>
#include <sys/socket.h>
```

```
#include <string.h>
#include <unistd.h>

#include <time.h>

#include "envoiPOST.h"

//partie fixe de la requete à adapter à vos besoins
	const char *messPOST="POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n"
	"Host: 192.168.1.11\r\n"
	"Content-Type: application/x-www-form-urlencoded\r\n"
	"Content-Length: ";

//prototypes de fonction
void attente(float temps);
int acquerirMesuresAnalogiques(int *,int *);

/**********************************/
void attente (float tps){
	//attente en seconde

	clock_t arrivee = clock()+(tps*CLOCKS_PER_SEC);
	while(clock()<arrivee);

}//attente()
/**********************************/
int acquerirMesuresAnalogiques(int *lmesure1, int *lmesure2){
	//acquiere les mesures analogiques sur la carte pcduino

		return 0;
}//acquerirMesuresAnalogiques
/**********************************/
int main_envoiPOST(int mesure1, int mesure2){

	printf("Entrez dans envoiPOST()\r\n");

	struct sockaddr_in adress_sock;//adresse socket avec type utilisé par connect()
	adress_sock.sin_family = AF_INET;//domaine IPV4 = AF_INET
	adress_sock.sin_port = htons(80);//port 80
	inet_aton("192.168.1.11",&adress_sock.sin_addr);// manipulation d'adresse : IP du serveur en local

	//creation de la partie variable de la requete contient les données
	char messData[50]="";
	int messLongData;
		//les données sont converties en char
	char cmesure1[4]="",cmesure2[4]="";
	sprintf(cmesure1,"%d",mesure1);sprintf(cmesure2,"%d",mesure2);
		//concatenation des mesures
	strcat(messData,"\r\n\r\nmesure1=");
	strcat(messData,cmesure1);
	strcat(messData,"&mesure2=");
	strcat(messData,cmesure2);
		//longeur des données en char
	messLongData = strlen(messData)-4; // -4 car on enlève les \r\n\r\n
	char cmessLongData[3]="";
	sprintf(cmessLongData,"%d",messLongData);

	//creation du message complet
```

```
	char messComplet[1024]="";
	strcat(messComplet,messPOST);
	strcat(messComplet,cmessLongData);
	strcat(messComplet,messData);
	printf("messComplet : %s\r\n",messComplet);
	//creation de la requete
	int monsocket = socket(PF_INET,SOCK_STREAM,0);
	int r=connect(monsocket,(struct sockaddr *)&adress_sock,sizeof(struct sockaddr_in));

	if (r!=1)//si connexion ok
	{
		//envoi de la requete POST
		write (monsocket,messComplet,strlen(messComplet));
	}//if

	return 0;
}//main_envoiPOST
```

Programme principal :

```
/*
 * main.c
 *
 *
 */


//#include <stdio.h>
//#include "analogin_sb.c"
//#include "gpio_sb.c"
#include "envoiPOST.c"


int main(int argc, char **argv)
{

	printf("Envoi POST start ...\r\n");
	while (1){

		printf("essai envoiPOST 2221\r\n");

		main_envoiPOST(33,133); //envoi des valeurs mesure1=33 et mesure2=133 en POST

		//allume LED attente 400ms
		//setPin(pinData[9], LOW); attente(0.4);
		//eteint LED
		//setPin(pinData[9],HIGH); attente(0.4);

		attente(10);

	}//while
	return 0;
}
```

Fichier envoiPOST().h

```
/*****************************************************************************
 * envoiPOST.h

modifie par sb
*
Header file for pcDuino GPIO example code.

26 Mar 2013 - Mike Hord, SparkFun Electronics

This code is beerware- if you find it useful, please buy me (or, for that
matter, any other SparkFun employee you met) a pint next time you meet us at
the local.

*****************************************************************************/

#ifndef __gpio_sb_h__
#define __gpio_sb_h__

// These are the paths and filenames of the files for GPIO access.
#define GPIO_MODE_PATH "/sys/devices/virtual/misc/gpio/mode/"
#define GPIO_PIN_PATH "/sys/devices/virtual/misc/gpio/pin/"
#define GPIO_FILENAME "gpio"

void writeFile(int fileID, int value);
void setPinMode(int pinID, int mode);
void setPin(int pinID, int state);
int main_envoiPOST();

// Make some aliases to make the code more readable.
#define HIGH '1'
#define LOW  '0'

#define INPUT  '0'
#define OUTPUT '1'
#define INPUT_PU '8'

#endif
```